1. (currently amended) A method for generating test cases that are converted to an abstract representation comprising:

using semantic analysis is used to convert the test cases to abstract representations;

providing rule-based generation of test cases from an abstract representation that includes application states, external interaction sequences and input data of test cases from data stores, wherein each application state is a set of application objects associated with a set of attributes and their values, or represents a runtime snapshot of an application under test which defines a context of external interaction;

validating generated test cases; and

converting the test cases to test scripts.

2. (original) The method of claim 1, wherein a data store is a relational database management system.

3. (original) The method of claim 1, wherein a data store is an XML database management system.

4. (original) The method of claim 1, wherein a data store is a file system.

5. (original) The method of claim 1, wherein an application state represents a runtime snapshot of application under test which defines the context of external interaction.

6. (original) The method of claim 5, wherein the application state includes a set of application objects, its attributes and attribute values.

7. (original) The method of claim 5, wherein the application states corresponding to a test case are arranged in a hierarchical manner.

8. (original) The method of claim 1, wherein the external interaction sequences represent events invoked by external agents on the application objects.

9 The method of claim 8, wherein the external agents are human agents or other software agents.

10. (original) The method of claim 8, wherein the interaction sequencing includes flow control structures for capturing sequential, concurrent, looping and conditional interactions.

11. (original) The method of claim 1, wherein the validation of generated test cases includes internal and external validation.

12. (original) The method of claim 11, wherein the internal validation ensures that the components of the test case definition, external interaction sequences and input data are consistent with each other and with an application object model.

13. (original) The method of claim 12, wherein an application object model is a metadata representation for modeling application under test.

14. (original) The method of claim 13, wherein the metadata representation includes object type definitions for application objects.

15. (original) The method of claim 13, wherein the metadata representation includes attribute definitions for each application object type.

16. (original) The method of claim 13, wherein the metadata representation includes definition of methods and events that are supported by each application object type.

17. (original) The method of claim 13, wherein the metadata representation includes definition of effects of events on an application state.

18. (original) The method of claim 14, wherein application object type definitions include additional categorization of each application object types into hierarchical, container and simple types.

19. (original) The method of claim 18, wherein the hierarchical object types are associated with an application state of its own, wherein application object types that can contain instances of other objects are termed container types.

20. (original) The method of claim 19, wherein the state associated with a hierarchical application object type is a modal application state or a nonmodal application state.

21. (original) The method of claim 20, wherein a modal application state restricts possible interactions to application object instances available within the current application state.

22. (original) The method of claim 17, wherein the effects of events on an application state capture one or more consequences of the event to the application state.

23. (original) The method of claim 22, wherein a consequence of an event is selected from, creation of a new object instance of a given type, deletion of an object instance of a given type, modification of attributes of an existing object instance and selection of an instance of an object type.

24. (original) The method of claim 23, wherein creation of a new instance of an object of type that is hierarchical results in creation of a new application state.

25. (original) The method of claim 23, wherein selection of an object instance of type that is hierarchical results in selection of the application state associated with that object instance.

26. (original) The method of claim 11, wherein the external validation validates the generated test case against the application metadata repository.

27. (original) The method of claim 26, wherein the application metadata repository contains definition of application objects and nature of their interactions within the application under test.

28. (original) The method of claim 26, wherein the external validation serves as a static verification test for the test cases.

29. (original) The method of claim 26, wherein the external validation increases productivity by pointing out invalid test cases.

30. (original) The method of claim 26, wherein the external validation increases productivity by pointing out inconsistencies in statically verifiable application behaviors.

31. (original) The method of claim 1, wherein the test scripts are test cases represented in a scripting language.

32. (original) The method of claim 31, wherein the scripting languages can be typed or untyped programming languages used for recording or authoring test cases.

33. (original) The method of claim 1, further comprising:

providing rules for selection of components of test case definition, external interaction sequences and input data; rules for data driven test case generation.

34. (original) The method of claim 33, wherein the selection rules are specified using query languages.

35. (original) The method of claim 34, wherein the query language is Structured Query Language (SQL).

36. (original) The method of claim 34, wherein the query language is XML Query (XQuery).

37. (original) The method of claim 34, wherein the query language is Application Programming Interface (API) called from code written in a programming language.

38. (original) The method of claim 34, wherein the use of query languages allows test cases to be generated from live customer data.

39. (original) The method of claim 33, wherein the data driven test case generation involves composing the test case as dictated by the input data.

40. (original) The method of claim 39, wherein the availability of multiple datasets for the input data will result in generation of multiple test cases or external interaction sequences repeated within a loop control structure for each dataset.

41. (original) The method of claim 39, wherein the availability of multiple datasets for a portion of the input data will result in the interaction sequences corresponding to this portion of input data repeated within a loop control structure.

42. (original) The method of claim 39, wherein each element of input data can be flagged valid or invalid.

43. (original) The method of claim 42, wherein the presence of validity flag in the input data that is different from the one corresponding the input data when the test cases was recorded or authored, results in the generator including appropriate interaction sequences for exception handling.

44. (original) The method of claim 1, further comprising:

converting test case from internal representation to a scripting language through language mapping.

45. (original) The method of claim 44, wherein the mapping is used to map external interactions captured as events on application object to appropriate statements in the scripting language.

46. (original) The method of claim 44, wherein more than one language mappings are provided at the same time.

47. (original) The method of claim 44, wherein the generated test case are converted to more than one scripting language at the same time.

48. (original) The method of claim 47, wherein generating test cases in multiple scripting language allows generation of test scripts for multiple test execution environments.

49. (currently amended) A computer system, comprising:

a processor:

a memory coupled to the processor, the memory storing rule-based generation of test cases from an abstract representation that includes application states, external

interaction sequences and input data of test cases from data stores to produce test cases;

logic that validates the test cases; and

logic for converting the test cases to test scripts, the logic using semantic analysis to convert the test cases to abstract representations that are one or more sets of application objects associated with a set of attributes and their values, or represents a runtime snapshot of an application under test which defines a context of external interaction.

50. (original) The system of claim 49, wherein the logic that validates the test cases provides that components of a test case definition, external interaction sequences and input data are consistent with each other and with an application object model.

51. (original) The system of claim 49, wherein the logic that validates the test cases is external validation logic.

52. (original) The system of claim 51, wherein the external validation logic includes validating a generated test case against an application metadata repository.

53. (original) The system of claim 49, further comprising:

logic for providing rules for selection of components of test case definition, external interaction sequences and input data; wherein and the rules are data driven test case generation.

54. (original) The system of claim 49, further comprising:

logic for providing data driven test case generation.

55. (currently amended) The system of claim 54, wherein the logic for providing data driven test case generation includes composing the test case as dictated by the input data.[[.]]